Introducing L Sharp, a Modern Lisp Dialect for the .NET Platform

Rob Blackwell, Active Web Solutions Ltd.

May 2005

Abstract

L Sharp [1] is a powerful Lisp-based scripting language for .NET. It uses a Lisp dialect similar to Arc [2] but tightly integrates with the .NET Framework [3] which provides a rich set of libraries.
L Sharp is free software distributed under the terms of the GNU General Public Licence [4]. You can download the latest release from www.lsharp.org.
This paper introduces L Sharp, explains its rationale and gives practical examples of its usage.

Keywords

Lisp, .NET, Arc, Scheme, Mono, Reflection.

Introduction

I work for Active Web Solutions, a software development and consultancy company based in the UK. We're working mostly (but not exclusively) with the Microsoft Platform and the C# (C Sharp) language. My development environment is:

Windows Server 2003
Internet Information Server (IIS 6)
Visual Studio .NET 2003
Visual Studio .NET 2005 Beta
NUnit Testing framework
Visual SourceSafe

The tools are okay, but the edit, compile, run, debug loop can be annoying and I miss the interactive development style of Lisp. Also, there is no really good scripting language. L Sharp aims to address these shortcomings by combining the power of a Lisp language with a really good, widely available class library, the .NET Framework.

Microsoft .NET

.NET has become something of a marketing buzzword at Microsoft. I use the term here in its original context as a language neutral execution environment created by Microsoft and subsequently standardised through ECMA (ECMA-335).

Languages that conform to the Common Language Infrastructure (CLI) all compile to Intermediate Language (IL) and all have access to the same class library. That means that it is possible to mix and match programming languages. There are some thirteen popular languages, the main ones being Visual Basic, C# and J#. All languages use the same Common Type System (CTS) and the same class library, "The Framework".

The .NET Framework is available on all current Microsoft operating systems (including handheld and mobile devices). Open source initiatives such as Mono provide implementations for other platforms including Linux, UNIX and Mac OS X.

The .NET Framework is a large, comprehensive class library providing a wide variety of functionality and application building blocks. Here is a small selection:

System.Collections
System.ComponentModel
System.Configuration
System.Data
System.Data.OracleClient
System.Data.SqlClient
System.Diagnostics
System.DirectoryServices
System.Drawing
System.EnterpriseServices
System.Globalization
System.IO
System.Management
System.Messaging
System.Net
System.Reflection
System.Resources
System.Runtime
System.Security
System.Threading
System.Web
System.Web.Mail
System.Web.Services
System.Xml

These cover just about all the building blocks you'd need to construct a typical enterprise application.

L Sharp

L Sharp is implemented entirely in C# as a .NET application.

The core language is provided as a class library assembly that can be loaded and embedded into any other .NET application. It can provide scripting for your application with only a few lines of code.

Most people start with the command line version which provides a familiar Lisp-like top loop.

```
C:\LSharp>lsharp
Welcome to L Sharp .NET, a powerful lisp-based scripting language for .NET.
Copyright (C) 2005 Rob Blackwell & Active Web Solutions.

This program is free software and is distributed under the terms of
the GNU General Public License.

For more information, see www.lsharp.org

Build 1.11.1962.14292 on Microsoft Windows NT 5.1.2600.0

> (writeline console "Hello World")
Hello World
null
>
```

This one line Hello World program is roughly equivalent to the following C# application which would need to be compiled before you could try it.

```
using System;

class Hello
{

        [STAThread]
        static void Main(string[] args)
        {
                Console.WriteLine ("Hello World");
        }


}
```

You will notice that WriteLine is not a part of the L Sharp language – the evaluation rule for L Sharp is that if the Car of the list isn’t a recognised L Sharp primitive, function or macro, it must be a method call to the underlying framework. This simple idea suddenly gives access to a massive library of functionality, with `(method object args)` being equivalent to `object.method(args);` in C#,

Here is a similar program, GUI style:

```
(reference "System.Windows.Forms")

(using "System.Windows.Forms")

(Show MessageBox "Hello World" "L Sharp")
```

The deployment units in .NET are called assemblies. The reference function loads an assembly (in this case System.Windows.Forms). The using directive permits the use of types in a namespace, such that you do not have to qualify the use of a type in that namespace. The last line above is the equivalent of MessageBox.Show ("Hello World", "LSharp") in C#.

Dialect

Microsoft has C# (a C-like derivative) and J#, a Java-like language. L Sharp was an obvious choice of name for a Lisp-like language.

In the same way that C# isn't C and J# isn't Java, L Sharp isn't Common Lisp; you can recognise the parentage but they aren't the same thing. L Sharp is tightly integrated with .NET and that means using the Common Type System.

When I started to experiment with L Sharp, I quite quickly established the mechanisms for creating Cons cells and Lists, as well as a very simple evaluator. With so much functionality available with no code (just by reflecting over and calling the framework), it was hard to decide on a Lisp dialect.

Common Lisp would have provided standardisation and perhaps a larger body of code, but I just didn't want or need all its facilities. I already had a type system, exceptions and a class library etc provided by the framework.

Scheme was a possibility, but I'd been reading Paul Graham's writings on Arc and it seemed to strike a chord with me. If L Sharp was going to be a modern Lisp implementation, then why not borrow Paul's ideas borne of many years' experience? I knew this would be controversial, but that's all part of the L Sharp experiment!

An RSS Viewer in 5 Lines

```
;;; Lists all the items from an RSS Feed

(reference "System.Xml")

(= news (new System.Xml.XmlDocument))

(call load news "http://www.theregister.co.uk/headlines.rss")

(foreach node (selectnodes news "/rss/channel/item/title")

    (prl (innertext node)))
```

I contend that this would be harder to write or at least, less readable in most other languages. The Lisp approach allowed me to write and try out the lines in the program incrementally; the framework gave me some very high level capability (loading and querying an XML document).

Implementation Notes

Because L Sharp is built in .NET, Lists can contain any .NET object. I had to rationalise the Lisp types with the Common Type System, so T becomes true and NIL becomes null.

L Sharp uses reflection to find objects and methods in the framework or any loaded class library. If L Sharp doesn't already include a feature that you need, it's easy to build that feature in another .NET language, load it and start using it straight away. Both the Read Table and the table of primitives can be changed, so you can override any of the built in L Sharp features and experiment with your own dialects.

Happily, the .NET framework already has a garbage collector so I didn't have to concern myself with memory management – Lists get garbage collected by the framework automatically.

The source code is simple, well commented, easily navigable and available as open source. It's easy to read, understand and extend.

Status

L Sharp was released early in 2005 and has already built up a small following. People are mostly using it for experimentation and small bits of “glue” code. Here are some typical current applications.

- Data import / export / mangling scripts; things that you may use Perl or Awk for on Linux can be done fairly easily on Windows with L Sharp.

- GUI Applications, replacing TK / TCL scripts

- Web site monitoring / testing tools.

- Experimentation with new Microsoft APIs such as Longhorn and Avalon,

Futures

Here are some of my objectives for future developments:

- Track what happens with Arc; continue to learn and borrow ideas. Full Arc compatibility may be a long term goal.

- The Mono open source .NET project is gaining significant momentum; L Sharp needs to ride that wave. L Sharp does work with Mono, but we need to do some more work to integrate it better and promote it within that community. I see this as the principal route to cross platform compatibility.

- Provide a compiler to generate standalone .NET assemblies. This could allow L Sharp applications to be more easily integrated with other .NET languages and developments, particularly for the provision of libraries. It should also give some performance increases.

- More attention needs to be given to an Integrated Development Environment. A simple prototype is included with the source distribution. Some people are using Emacs but we don’t yet have SLIME support. A Visual Studio add-in would make L Sharp more accessible to the .NET community.

- It should be possible to use L Sharp as a web application scripting language using ASP.NET.

- L Sharp needs some larger applications to prove that it is ready for real world use.

- I encourage more input from The Lisp Community. I need more people to try L Sharp, give me feedback and hopefully help to extend and grow the language.

References

[1] The L Sharp Home Page http://www.lsharp.org

[2] Arc is a new dialect of Lisp by Paul Graham.
http://www.paulgraham.com/arc.html

[3] Microsoft .NET Home Page http://www.microsoft.com/net/

[4] The GNU General Public License http://www.gnu.org/copyleft/gpl.html